Automatic Linear Correction of Rounding Errors

Philippe Langlois

INRIA

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE

Automatic Linear Correction of Rounding Errors

Philippe LANGLOIS

No 3828

Decembre 1999

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--3828--FR+ENG

Automatic Linear Correction of Rounding Errors

Philippe LANGLOIS *

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n ° 3828 — Decembre 1999 — ?? pages

Abstract: A new automatic method to correct the first-order effect of floating point rounding errors on the result of a numerical algorithm is presented. A correcting term and a confidence threshold are computed using automatic differentiation, computation of elementary rounding error and running error analysis. Algorithms for which the accuracy of the result is not affected by higher order terms are identified. The correction is applied to the final result or to sensitive intermediate results. The properties and the efficiency of the method are illustrated with a sample numerical example.

Key-words: Automatic error analysis, rounding error, floating point arithmetic.

(Résumé : tsvp)

* Email: `Philippe.Langlois@ens-lyon.fr`. Part of this research was done while the author was visiting the Department of Mathematics at the University of Manchester during his C.R.C.T. period.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Correction Linéaire Automatique des Erreurs d'Arrondi

Résumé : Une nouvelle méthode automatique de correction de l'effet d'ordre un des erreurs d'arrondi introduites par l'arithmétique flottante est présentée. Un terme correctif et un intervalle de confiance sont calculés par différentiation automatique, calcul des erreurs d'arrondi élémentaires et analyse de type *running error*. Les algorithmes dont la précision du résultat n'est pas affectée par des termes d'ordre supérieur sont identifiés. La correction est appliquée au résultat final ou à des résultats intermédiaires sensibles. Les propriétés et l'efficacité de la méthode sont illustrées par un exemple numérique adéquat.

1 Introduction

Improving the accuracy and the stability of numerical software is one of the current challenges in scientific computing. The rounding errors introduced by floating point arithmetic contribute to the inaccuracy of the computed results and to the instability of some numerical algorithms. In this paper, we focus on the propagation of rounding errors. Other sources of error such as errors in the data and the use of approximation schemes are not considered here.

The control of the propagation of rounding errors and the improvement of the final accuracy have been extensively studied for more than 40 years. Even so, new results and recent books in the domain are still numerous [3], [5], [12], [31].

Rounding error analysis becomes complex and tedious for large algorithms, and of course, for the software which implements them. Therefore, some automatic approaches that use the computer to correct its own deficiencies have been developed. We present a brief review of these automatic methods in Section 2.

We define the *elementary rounding error* as the error introduced by each floating point arithmetic operation, and the *global forward (rounding) error* as the accumulated effect of the elementary rounding errors on the final result.

Some automatic methods rely on the computation of the partial derivatives of the global forward error with respect to the elementary rounding errors. These *differential methods* use automatic differentiation techniques that provide a general application and an efficient implementation [14]. Differential methods are generally used to compute a bound for the global forward error. However, as described in Section 3, such computed bounds may be too large to be useful.

In this paper, we present the *CENA method* that provides an automatic linear correction of rounding errors. CENA is the French acronym of *Correction des Erreurs Numériques d'Arrondi*, that means Correcting Numerical Rounding Errors. This new differential method does not attempt to compute a bound for the global forward error. The CENA method actually *corrects* the computed result with the first-order terms of the global forward error with respect to the elementary rounding errors.

Such a linear correction is suitable when the global forward error is dominated by the first-order terms, for example in the *linear algorithms* we define in Section 3. Even if complex algorithms are not linear, the accuracy of computed solutions may rely on linear parts of the algorithms. Thus, the CENA method can be used to correct final results or sensitive intermediate variables and improve the accuracy of the results and the stability of the algorithm.

The computed linear correction term may also suffer from an approximation error and rounding errors. In this case, the remaining difference between the corrected result and the exact result is measured by a *residual error*. The CENA method computes a bound of this residual error that provides a confidence threshold associated with the corrected result.

The CENA method relies on the computation of the elementary rounding errors and the partial derivatives of the computed result with respect to these elementary rounding errors. These two computations are respectively presented in Sections 4 and 5.

Langlois and Nativel have given an introductory description of this approach in [18] (in French) and [19]. Examples of final result correction are proposed in Nativel's PhD dissertation [27]. The aim of this paper is to present a complete and definitive description of the CENA method. Using Wilkinson's running error analysis, we also derive here a new dynamic bound for the residual error that improves the previous *a priori* result given in [18]. The properties of the CENA method are illustrated throughout this paper with the evaluation of a sample function. Significant applications of the CENA method to final and intermediate sensitive results are proposed in [17].

2 Automatic Rounding Error Methods

Two parameters characterize automatic methods for rounding errors control.

- The *space* within the result of the rounding error analysis is expressed.
- The *deterministic* or *stochastic* properties held by the approach.

Considering a numerical algorithm as a mapping f of a data space $\mathcal{D}$ to a result space $\mathcal{R}$, a *forward* method estimates or bounds the loss of accuracy $\|\widehat{f}(X) - f(X)\|_{\mathcal{R}}$ of the computed result $\widehat{f}(X)$ for data $X \in \mathcal{D}$ and a norm $\|\cdot\|_{\mathcal{R}}$ on $\mathcal{R}$.

The *backward* approach interprets the final inaccurate result $\widehat{f}(X)$ as exact for the same mapping f, but applied to perturbed data $X + \Delta X$. Thus, $\widehat{f}(X) = f(X + \Delta X)$, and the backward analysis consists in bounding the backward error $\|\Delta X\|_{\mathcal{D}}$ in the data space.

Assuming f continuously differentiable, a *condition number* measures the sensitivity $\partial f / \partial X$ in a neighborhood of the data X. Condition numbers depend on the choice of the norms $\|\cdot\|_{\mathcal{D}}$ and $\|\cdot\|_{\mathcal{R}}$ in the data and result spaces.

Forward and backward approaches are linked by the first-order inequality

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}$$

which holds for regular solutions and consistently defined errors and condition number. This relation can be generalized to singular solutions using a conjecture, proposed in [5], that relies on regularity properties of f.

We distinguish forward and backward approaches according to the space within the conclusions of the error analysis are expressed. That is, respectively, the result space $\mathcal{R}$ for the forward approaches and the data space $\mathcal{D}$ for the backward. Distinction between deterministic and stochastic methods is less systematic as stochastic properties could be introduced at different levels of the method. Therefore, we consider a method to be *deterministic* if no

stochastic hypothesis is assumed; otherwise we consider it to be *stochastic*. According to this characterization, we briefly review the main automatic rounding error methods.

Introduced in 1966 by Moore [26] and widely developed and automated by Kulisch and his colleagues [16], interval analysis is a forward deterministic approach. The propagation of rounding errors is simulated using interval arithmetic. Each interval contains the exact solution and its range quantifies the effect of rounding errors propagation. The classical drawback is the growth of the interval result that tends to overestimate the inaccuracy of the computed result. However, optimal interval results may be computed using the maximum-quality inner product, or iterative refinement when the problem is equivalent to a contractive fixed-point equation, or more generally, other techniques that depend on the problem and the algorithm [2]. Numerous libraries and programming language extensions that implement interval analysis are available [1].

Wilkinson's running error analysis is another forward deterministic method proposed in 1971 by [28] but was applied by Wilkinson much earlier [37]. Rudimentary but both general and easy to implement, this method is more widely known since it is described in [12]. The absolute value of the results of each arithmetic operation are added in an accumulation variable. A *running error bound* of the global forward error is therefore derived by the product of this variable with the arithmetic precision.

The CESTAC method, introduced in 1974 by La Porte and Vignes [32] and developed by Vignes and his colleagues, is a forward stochastic method [36]. The model for the elementary rounding errors is a normal distribution which depends on properties of the arithmetic rounding mode. This method simulates such elementary rounding errors and provides the number of significant digits from a first-order model of the elementary rounding error propagation. The stochastic arithmetic formalizes the underlying hypothesis and the mathematical founding of this approach [7]. The software tool CADNA implements the CESTAC method in Ada and Fortran 90 [6].

Since 1988, Chaitin-Chatelin and her colleagues have been working on a backward statistical approach. The mathematical formulation of this approach is described in [5]. The numerical stability with respect to the rounding errors or initial data is analyzed according to the backward approach. The associated implementation, PRECISE, provides statistical estimations of condition numbers, backward errors and regularity orders, as well as statistical sensitivity analysis. These tools are mainly applied to solve linear systems, eigenvalues problems and the determination of polynomial roots. Similar approaches are proposed in [9] and [10].

Writing the global forward error as the first-order Taylor expansion with respect to the elementary rounding errors, is known since at least 1964 [11]. In 1972–1973, Linnainmaa and Miller both proposed an automatic use of this Taylor expansion and consequently defined

Table 1: Main automatic rounding error methods.

Method	Analysis	Model	Main references
Interval Arithmetic	forward	deterministic	[26],[1]
Running Error Analysis	forward	deterministic	[28],[37]
CESTAC	forward	stochastic	[32],[31]
PRECISE	backward	stochastic	[5]
Absolute Error Analysis	backward	deterministic	[25]
Relative Error Analysis	forward, backward	deterministic	[20]
Error Linearization	forward	stochastic	[23]
CENA	forward	deterministic	

the first software tools for automatic differentiation [21], [24]. These differential methods are stochastic or deterministic according to the elementary rounding error model : stochastic in [23], deterministic in [20], [25] and both in [14]. Both forward and backward error analyses were derived from these differential methods. The first-order Taylor expansion directly yields the forward error analysis. Backward error analysis is derived comparing the first-order forward analysis of rounding errors and data perturbations. This perturbation analysis is automated computing the derivatives of the solution with respect to the data [25]. Experimental software, mostly free and in Fortran 77, implement these developments.

The CENA method presented in this paper is a forward and deterministic approach. Both first-order approximation of the global forward error with respect to the elementary rounding errors and automatic differentiation are applied. Elementary rounding errors are neither bounded nor described by a stochastic model, but computed. The first-order term of the global forward error with respect to the elementary rounding errors is computed as a correcting factor to the computed result.

Linnainmaa has proven that the first-order approximation of the global forward error is an efficient tool for experimental analysis of the numerical stability of algorithms [23]. The CENA method extends Linnainmaa's error linearization method providing the computation of a correcting factor of the global forward error. This correction is based on the numerous results on elementary error computation obtained between 1965–1975 by several authors, particularly Dekker [8], Kahan, Knuth [15] and Pichat [30].

All these approaches are *a posteriori* methods, that is, the assessed algorithm or software proceeds and jointly provides the computed result and the analysis. Table 1 summarizes this classification. To complete this review of the main automatic methods for rounding control, we mention rather different approaches based on symbolic computation as proposed in [34].

3 Rounding Errors and Linear Algorithms

In this section, we define the notation used in the paper and then the principles of the CENA method.

3.1 Floating Point Environment

Let $\mathbb{F}$ be the set of normalized floating point numbers with p digits of mantissa in base b. Elementary floating point operations that correspond to arithmetic operations $(+, -, \times, /, \sqrt{})$ are defined on $\mathbb{F}$, and evaluated expressions are denoted $fl(\cdot)$. Computed quantities wear a hat as in [12] for example.

Using Dekker's definitions, we assume that the floating point arithmetic is optimal or at least faithful [8]. A *faithful* arithmetic is such that a computed result $\widehat{z}$ is equal to the exact result z when $z \in \mathbb{F}$, and one of the two adjacent elements in $\mathbb{F}$ that enclose the exact result z otherwise. An *optimal* arithmetic is a faithful arithmetic such that the computed result $\widehat{z}$ is the nearest element in $\mathbb{F}$ of the exact result z. A tie-breaking strategy is supposed to be defined when there are two nearest elements. The rounding unit of $\mathbb{F}$ is $\mathbf{u} = \mathsf{b}^{1-p}$ for a faithful arithmetic and $\mathbf{u} = \mathsf{b}^{1-p}/2$ for an optimal one.

Thus, a faithful or optimal arithmetic satisfies the standard model of floating point arithmetic [12]. For $x, y \in \mathbb{F}$ and $\circ \in \{+, -, \times, /, \sqrt{}\}$, we define $z = x \circ y$ and $\widehat{z} = fl(x \circ y)$. Then,

$$\widehat{z} = z(1+\zeta), \quad |\zeta| \leq \mathbf{u}, \tag{1}$$

and similarly,

$$\widehat{z} = z/(1+\zeta), \quad |\zeta| \leq \mathbf{u}. \tag{2}$$

3.2 Linear Correction

Let f be a function of n real variables and $fl(f) = \widehat{f}$ its numerical evaluation on $\mathbb{F}$. For simplicity, we assume that f is a real function — a vector function will be considered componentwise. For $X = (x_1, \ldots, x_n)$ belonging to $\mathbb{F}$, consider the computation of $x_N = f(X)$ that returns $\widehat{x}_N$. The global forward error is $\widehat{x}_N - f(X)$. The computation of each intermediate variable $\widehat{x}_k$ introduces an elementary rounding error δ_k for $k = n+1, \ldots, N$. For $\widehat{x}_i$, $\widehat{x}_j$ and $\widehat{x}_k$ in $\mathbb{F}$ such that $1 \leq i \leq j < k \leq N$ and $\circ \in \{+, -, \times, /, \sqrt{}\}$,

$$\begin{aligned} \widehat{x}_k &= fl(\widehat{x}_i \circ \widehat{x}_j) \\ &= \widehat{x}_i \circ \widehat{x}_j + \delta_k \qquad (3) \\ &= x_k + \delta_k, \qquad (4) \end{aligned}$$

where $\widehat{x}_j \neq 0$ when $\circ = /$, and $\widehat{x}_i = 0$, $\widehat{x}_j \geq 0$ when $\circ = \sqrt{}$.

Table 2: Examples of linear bound weakness.

Expression	Computed value $\widehat{x}$	Actual $\lvert \widehat{x} - x \rvert$	Linear bounding of $\lvert \widehat{x} - x \rvert$
$(2^{50}+1)-1$	2^{50}	0	2^{27}
$(2^{25})^2 - 2^{50}$	0	0	2^{27}

So, the numerical evaluation of $f(X)$ on $\mathbb{F}$ is $\widehat{f}(X,\delta)$, that is, a function of the data X and the elementary rounding errors $\delta = (\delta_{n+1}, \ldots, \delta_N)$. A first-order approximate of the global rounding error with respect to the elementary rounding errors is

$$\Delta_L = \sum_{k=n+1}^{N} \frac{\partial \widehat{f}}{\partial \delta_k}(X,\delta) \cdot \delta_k \,. \tag{5}$$

As $f(X) = \widehat{f}(X,0)$, the global forward rounding error is

$$\widehat{f}(X,\delta) - f(X) = \Delta_L - E_L, \tag{6}$$

where E_L is the *linearization error* associated with Δ_L.

This approximate Δ_L is computable when the partial derivatives $\partial \widehat{f}/\partial \delta_k$ and the elementary rounding errors δ_k are known. Numerous stochastic models for δ_k have been tried [14], [23], [35], though elementary rounding errors are not random but fixed for given arithmetic and values.

Bounding δ_k by $\mathbf{u}$ in the deterministic approaches, as in [14], sometimes yields unsuitable bounds for $|\Delta_L|$. Indeed, the compensation of elementary rounding errors and exactly computed results are not taken into account when such bounds are derived. In Table 2, we present two examples that respectively illustrate these limitations in IEEE-754 single precision ($\mathbf{u} = 2^{-24}$) [13].

In order to avoid these drawbacks, we propose not to bound, but to *compute* Δ_L. Hence, we compute the *corrected result* $\overline{f(X)}$ defined as

$$\overline{f(X)} = fl\left(\widehat{f}(X,\delta) - \widehat{\Delta_L}\right), \tag{7}$$

with

$$\widehat{\Delta_L} = \Delta_L + E'_C,$$

where E'_C is the error introduced by the computation of relation (5) in $\mathbb{F}$. The final subtraction of the correction defined by relation (7) introduces an elementary rounding error

$$\delta = \overline{f(X)} - (\widehat{f}(X,\delta) - \widehat{\Delta_L}),$$

such that

$$|\delta| \leq \mathbf{u}\,|\overline{f(X)}|. \tag{8}$$

Hence, the *computing error* associated with the corrected result $\overline{f(X)}$ is

$$E_C = E'_C - \delta. \tag{9}$$

Finally, the *residual error* between the corrected result and the exact result is

$$\overline{f(X)} - f(X) = -(E_L + E_C). \tag{10}$$

The smaller is the residual error, the more efficient is the correcting method.

In next Section 3.3, we study the linearization error E_L. In Section 3.4, we derive a computable bound of the computing error E_C that is the residual error for a particular class of algorithms, the *linear algorithms*. Then, we discuss the attainable accuracy of the corrected result and introduce the notion of sensitive intermediate result in Section 3.5.

3.3 The Linearization Error E_L and Linear Algorithms

Higher than first-order effects of the elementary rounding errors may contribute to the inaccuracy of a computed result.

For example, let $\widehat{f}_1$ be the evaluation of $f(x,y,z) = x^2 - y^2 - z^2$ with Algorithm 1.

Algorithm 1 computes $f(x,y,z) = x^2 - y^2 - z^2$ introducing second-order rounding error s.

$x_1 = x, \quad x_2 = y, \quad x_3 = z$
$x_4 = x_1 + x_2$
$x_5 = x_1 - x_2$
$x_6 = x_4 \times x_5$
$x_7 = x_3 \times x_3$
$x_8 = x_6 - x_7$
$\widehat{f}_1(x,y,z) = x_8$

Relation (4) yields

$$\widehat{f}_1(x,y,z) - f(x,y,z) = \delta_4(x-y) + \delta_5(x+y) + \delta_6 + \delta_7 + \delta_8 + \delta_4\delta_5.$$

Then, computing $f(2^{25}, 1, 2^{25}) = -1$ in IEEE-754 single precision ($\mathbf{u} = 2^{-24}$) gives

$$\widehat{f}_1(2^{25}, 1, 2^{25}) = 0,$$

as

$$\delta_4 = -1, \quad \delta_5 = 1 \quad \text{and} \quad \delta_6 = \delta_7 = \delta_8 = 0.$$

In this case, both the first and second-order terms significantly contribute to the global forward error.

The order and terms that are significant in the development of the global forward error with respect to the elementary rounding errors, depend on the algorithm and the data. A general criterion to stop computing higher order terms is difficult to define. The *a priori* choice of a maximal order such that higher order terms are not considered may involve to neglect significant terms for the global rounding error. Such terms are here taken into account in the linearization error E_L defined by relation (6).

An accurate bound of E_L is also difficult to compute. Nevertheless, the following linear algorithms satisfies the interesting property $E_L = 0$.

Definition 1 *A* linear algorithm on $\mathbb{F}$ *is an algorithm that only contains the operations* $\{+, -, \times, /, \sqrt{}\}$ *and such that*
- *every multiplication* $fl(\widehat{x}_i \times \widehat{x}_j)$ *satisfies* $\widehat{x}_i = x_i$ *or* $\widehat{x}_j = x_j$, *and*
- *every division* $fl(\widehat{x}_i / \widehat{x}_j)$ *satisfies* $\widehat{x}_j = x_j$, *and*
- *every square root* $fl\left(\sqrt{\widehat{x}_i}\right)$ *satisfies* $\widehat{x}_i = x_i$.

Linear algorithms are such that the identified operands $\widehat{x}_k$ suffer from no previous rounding error. Constants and data of any implemented algorithm are such $\widehat{x}_k$. According to their numerical value, some computed quantities may also satisfy $\widehat{x}_k = x_k$. Even if the final result $\widehat{x}_N$ is not computed with a linear algorithm, some intermediate variables may be returned by linear parts of the algorithm.

Some basic algorithms for scientific computing are linear algorithms. For example, we mention the natural order summation algorithm of n real numbers, the inner product of two vectors and therefore most of the BLAS routines, the polynomial evaluation using Horner's method and the resolution of triangular linear system.

We compute the previous function $f(x, y, z)$ with the following linear algorithm that returns $\widehat{f_2}$.

Algorithm 2 is a linear algorithm to compute $f(x, y, z) = x^2 - y^2 - z^2$.

$x_1 = x, \quad x_2 = y, \quad x_3 = z$
$x_4 = x_1 \times x_1$
$x_5 = x_2 \times x_2$
$x_6 = x_4 - x_5$
$x_7 = x_3 \times x_3$
$x_8 = x_6 - x_7$
$\widehat{f_2}(x, y, z) = x_8$

We have

$$\widehat{f_2}(x,y,z) - f(x,y,z) = \sum_{k=4}^{8} \delta_k.$$

The global forward error of the linear algorithm $\widehat{f_2}$ is a linear function of the elementary rounding errors. This property holds for linear algorithms.

Theorem 1 *Let $\widehat{x}_N$ be the computed result of $f(X)$ by a linear algorithm on $\mathbb{F}$ for $X = (x_1, \ldots, x_n) \in \mathbb{F}^n$. The global forward rounding error $\widehat{x}_N - f(X)$ is a linear function of the elementary rounding errors $\delta = (\delta_{n+1}, \ldots, \delta_N)$.*

proof (Main steps) For each elementary operation and $k = n+1, \ldots, N$, we prove by induction that the global rounding error in $\widehat{x}_k$ is a linear function of the elementary rounding errors δ_r $(n+1 \leq r \leq k)$.

A non-linear propagation of the elementary rounding error δ_r may appear when $\widehat{x}_r$ is one of the two operands of a multiplication, the divisor of a division or the operand of a square root. For the multiplication and $n+1 \leq s \leq r < k$,

$$\begin{aligned} \widehat{x}_k &= \widehat{x}_r \times \widehat{x}_s + \delta_k \\ &= (x_r + \delta_r) \times (x_s + \delta_s) + \delta_k \\ &= x_k + \delta_k + \delta_r x_s + \delta_s x_r + \delta_r \delta_s. \end{aligned}$$

From the linear algorithm definition, we assume for example that $\delta_r = 0$. The non-linear term $\delta_r \delta_s$ vanishes, and the global rounding error $\widehat{x}_k - x_k$ is linear with respect to δ_s and δ_k. The induction is initialized since the data $X = (x_1, \ldots, x_n) \in \mathbb{F}^n$. The division and square root cases are similar and presented in [19].
□

For a given computation, a linear algorithm provides a linear global forward error. Of course, this global forward error is not necessary smaller than the error produced by another non-linear algorithm. For example, the two previous computations of $\widehat{f_1}$ and $\widehat{f_2}$ give the same forward error for $(x,y,z) = (2^{25}, 1, 2^{25})$. The linear algorithm generates a global forward error that only consists in first-order terms with respect to the elementary rounding errors.

As $E_L = 0$, the two following corollaries for linear algorithms and the corrected result derive from relation (10). We recall that $\overline{f(X)} = fl\left(\widehat{f}(X,\delta) - \widehat{\Delta_L}\right)$ and the computing error E_C introduced by the computation of $\widehat{\Delta_L} = fl\,(\Delta_L)$ is defined by relation (9).

Corollary 1 *When $f(X)$ is computed with a linear algorithm, the corrected result $\overline{f(X)}$ only suffers from the computing error E_C,* i.e.*,*

$$\overline{f(X)} = f(X) - E_C.$$

The CENA method computes the linear correcting factor $\widehat{\Delta_L}$ and then the corrected result $\overline{f(X)}$. Let $\widehat{f}(X,\delta)$ be the computed result of a linear (part of) algorithm. The corrected result $\overline{f(X)}$ does not suffer from the first-order effect of the elementary rounding errors. Hence, we have for the previous example

$$\overline{f_2(2^{25},1,2^{25})} = -1 = f(2^{25},1,2^{25}).$$

Corollary 2 *Let B_{E_C} be a bound for E_C, that is, $|E_C| \leq B_{E_C}$. The exact result $f(X)$ and the corrected result $\overline{f(X)}$ of a linear algorithm are such that*

$$f(X) \in \mathcal{I}_{E_C} = [\overline{f(X)} - B_{E_C}, \overline{f(X)} + B_{E_C}].$$

The B_{E_C} bound of the rounding errors introduced by the correction process provides a confidence threshold for the corrected result $\overline{f(X)}$. Let us assume that B_{E_C}, and hence the interval $\mathcal{I}_{E_C}$, are computed (see next section). From Corollary 2, $f(X)$ belongs to $\mathcal{I}_{E_C}$ (but is unknown). Thus, the corrected value $\overline{f(X)}$ and the confidence threshold B_{E_C} could be used to control the accuracy of the original computed result $\widehat{x}_N$. Comparing $\widehat{x}_N = \widehat{f}(X,\delta)$ with $\mathcal{I}_{E_C}$ range may provide an estimate of the number of correct digits in the computed result.

In the previous example, we compute $B_{E_C} = 7.152 \times 10^{-7}$ in optimal IEEE-754 single precision arithmetic, *i.e.*, $\mathbf{u} = 2^{-24} \approx 5.96 \times 10^{-8}$. As the corrected result $\overline{f_2(2^{25},1,2^{25})} = -1.0$, the exact result $f(2^{25},1,2^{25}) = -1.0 \pm 7.152 \times 10^{-7}$. So we prove that the initial computed value $\widehat{f}_2(2^{25},1,2^{25}) = 0.0$ has no significant digit.

3.4 The Computing Error E_C and its Bounding

The correction $fl\left(\widehat{f} - \widehat{\Delta_L}\right)$ relies on the computation of the inner product Δ_L of the elementary errors δ_k and the partial derivatives $\partial\,\widehat{f}/\partial\delta_k$ $(n+1 \leq k \leq N)$. As these computations all suffer from rounding errors, is the correcting term $\widehat{\Delta_L}$ accurate enough to improve the accuracy of the computed result ?

From Section 3.2, we recall that the computing error is $E_C = E'_C - \delta$, where δ is the elementary rounding error due to the last subtraction that satisfies $|\delta| \leq \mathbf{u}\,|\overline{f(X)}|$. We discuss the accuracy of this subtraction in the next Section 3.5.

Langlois and Nativel provide a theoretical approximate bound for $E'_C = \widehat{\Delta_L} - \Delta_L$ that derives from *a priori* inequalities [18]. However, this bound suffers from the characteristic weakness of *a priori* results: the actual elementary errors are not taken into account as all these errors are bounded by the worst one. The *a priori* bound also increases as the number of intermediate variables that could be very large in applications. So, the computation of this *a priori* bound for E'_C may return large bounds and suffer from overflow.

We propose here to compute a sharper bound for the computation error E_C using Wilkinson's running error analysis recalled in Section 2.

Algorithm 3 is a model of $\widehat{\Delta_L}$ computation using relation (5).

$S_n = 0$
for $k = n+1$ **to** N **do**
  $U_k = A(u_k)$
  $V_k = B(v_k)$
  $P_k = U_k \times V_k$
  $S_k = S_{k-1} + P_k$
end for
$\widehat{\Delta_L} = S_N$

The computation of the correcting factor $\widehat{\Delta_L}$ with relation (5) is described by Algorithm 3 where $u_k = (\partial \widehat{f}/\partial \delta_k)(X, \delta)$ and $v_k = \delta_k$ $(k = n+1, \ldots, N)$. The functions A and B are introduced to take into account the rounding errors due to the previous computations of the partial derivatives $\partial \widehat{f}/\partial \delta_k$ and the elementary errors δ_k. Let us assume that global rounding error bounds on U_k and V_k are computable and such that

$$\widehat{U}_k - u_k = \mu_k \quad \text{with} \quad |\mu_k| \leq \mathbf{u}\,\alpha_k \quad (\mu_k, \alpha_k \in \mathbb{R}), \tag{11}$$

and

$$\widehat{V}_k - v_k = \nu_k \quad \text{with} \quad |\nu_k| \leq \mathbf{u}\,\beta_k \quad (\nu_k, \beta_k \in \mathbb{R}). \tag{12}$$

Such bounds α_k and β_k are proposed in Sections 5 and 4.

Applying relations (1) and (2) to $\widehat{P_k}$ and $\widehat{S_k}$ computed by Algorithm 3 yield

$$\widehat{P}_k = \frac{\widehat{U}_k\,\widehat{V}_k}{1+\pi_k}, \quad |\pi_k| \leq \mathbf{u},$$

and

$$(1+\sigma_k)\,\widehat{S}_k = \widehat{S}_{k-1} + \widehat{U}_k\,\widehat{V}_k - \pi_k\,\widehat{P}_k, \quad |\sigma_k| \leq \mathbf{u}.$$

Relations (11) and (12) give

$$\widehat{U}_k\,\widehat{V}_k = u_k v_k + \mu_k\,\widehat{V}_k + \nu_k\,\widehat{U}_k - \mu_k\nu_k.$$

With the error $e_k = \widehat{S}_k - S_k$ in relation (3.4) and as $S_k = S_{k-1} + u_k v_k$, we have

$$e_k = e_{k-1} - \pi_k\,\widehat{P}_k - \sigma_k\,\widehat{S}_k + \mu_k\,\widehat{V}_k + \nu_k\,\widehat{U}_k - \mu_k\nu_k.$$

So, a bound that only depends on computable quantities is

$$|e_k| \leq |e_{k-1}| + \mathbf{u}\,(|\,\widehat{P}_k| + |\,\widehat{S}_k| + \alpha_k|\,\widehat{V}_k| + \beta_k|\,\widehat{U}_k| + \mathbf{u}\,\alpha_k\beta_k).$$

We write the bound $|e_k| = e_k^{(1)} + e_k^{(2)}$, where $e_k^{(1)}$ and $e_k^{(2)}$ respectively represent the first and second-order terms in $\mathbf{u}$ of $|e_k|$. Since $e_n = 0$, Algorithm 4 is the transformation of Algorithm 3 to compute both the correcting factor $\widehat{\Delta_L}$ and a running error bound $B_{E'_C}$ for E'_C.

Algorithm 4 computes both $\widehat{\Delta_L}$ and a running error bound $B_{E'_C}$ for E'_C.

$S_n = 0, \quad e_n^{(1)} = 0, \quad e_n^{(2)} = 0$
for $k = n+1$ **to** N **do**
  $U_k = A(u_k)$
  $V_k = B(v_k)$
  $P_k = U_k \times V_k$
  $S_k = S_{k-1} + P_k$
  $e_k^{(1)} = e_{k-1}^{(1)} + |P_k| + |S_k| + \alpha_k \times |V_k| + \beta_k \times |U_k|$
  $e_k^{(2)} = e_{k-1}^{(2)} + \alpha_k \times \beta_k$
end for
$\widehat{\Delta_L} = S_N$
$B_{E'_C} = \mathbf{u} \times (e_N^{(1)} + \mathbf{u} \times e_N^{(2)})$

Knowing the corrected result $\overline{f(X)}$ in relation (8), a bound B_{E_C} for the computing error $E_C = E'_C - \delta$ introduced by the correcting process is

$$B_{E_C} = \mathbf{u}[(e_N^{(1)} + \overline{f(X)}) + \mathbf{u}e_N^{(2)}]. \tag{13}$$

Hence, the immediate modification of Algorithm 4, replacing its last instruction by relation (13), provides the computation of the correcting term $\widehat{\Delta_L}$ and the associated running error bound B_{E_C}.

In practice, a first order running error bound is computed neglecting second-order terms $e_k^{(2)}$ $(k = n+1, \ldots, N)$. Numerical experiments confirm that running error bounds are more accurate than *a priori* bounds. Algorithm 4 provides about 100 times sharper bounds than the previous result from [18]. This bound improves the previously pointed out assessment of the computed result accuracy that Corollary 2 provides. However, pessimistic bounds and overflow risks are not definitely avoided as a running error bound is also a summation of absolute values which necessarily increases as the computation proceeds.

3.5 Accuracy Limitation of the Corrected Result

The best corrected result $\overline{f(X)}$ satisfies

$$\overline{f(X)} = (1+\epsilon)f(X), \quad \epsilon = 0 \text{ when } f(X) \in \mathbb{F} \text{ and } |\epsilon| \leq \mathbf{u} \text{ otherwise.} \tag{14}$$

The actual corrected result does not satisfy this property as it suffers from the errors E_L and E_C (see Corollary 2).

The initial accuracy of the computed result $\widehat{f}(X)$ and the precision $\mathbf{u}$ also limit the accuracy of the corrected result $\overline{f(X)}$. The final subtraction $fl\left(\widehat{f}(X) - \widehat{\Delta_L}\right)$ may suffer from severe cancellation and magnifies previous errors in $\widehat{f}(X)$ and $\widehat{\Delta_L}$. The exact correction $\Delta_L = \widehat{f} - f$ yields in $\mathbb{F}$ the best correcting factor $\widehat{\Delta_L} = fl(\Delta_L)$, *i.e.*, $\widehat{\Delta_L} = \underset{\Delta \in \mathbb{F}}{min}\{|f - fl\left(\widehat{f} - \widehat{\Delta_L}\right)|\}$. The worst cancellation in $fl\left(\widehat{f}(X) - \widehat{\Delta_L}\right)$ appears when $\mathbf{u}|\widehat{f}(X)| > |f(X)| > 0$: in precision $\mathbf{u}$, $\widehat{\Delta_L} = \widehat{f}(X)$ and $\overline{f(X)} = 0$. This corrected result does not satisfy relation (14) for $f(X) \neq 0$. Nevertheless, the corrected result is more accurate than the initially computed result as $|\widehat{f}(X)| \gg |f(X)| > |\overline{f(X)}| = 0$.

For example, let us consider the IEEE-754 single precision evaluation of $g(2^{50}, 2^{25}, 1)$, with $g(x,y,z) = z^2$ but computed as

$$\widehat{g}(x,y,z) = x^2 - y^2 - x^2 + y^2 + z^2.$$

Evaluating $\widehat{g}$ from the left to the right insures $E_L = 0$. We have $g(2^{50}, 2^{25}, 1) = 1$ and $\widehat{g}(2^{50}, 2^{25}, 1) = 2^{50}$. The global error $\Delta_L = 2^{50} - 1$ yields a correcting factor $\widehat{\Delta_L} = 2^{50}$ in finite precision $\mathbf{u} = 2^{-24}$. With such a Δ_L, the corrected result would be $\overline{g(2^{50}, 2^{25}, 1)} = 0$, that is not an accurate correction of the exact value 1.

A natural way to avoid this limitation is to reduce the global error in $\widehat{f}(X)$ before it becomes too important to be corrected with the final subtraction $fl\left(\widehat{f}(X) - \widehat{\Delta_L}\right)$. This may be achieved by *correcting* some well-chosen *intermediate results* in the computation of $\widehat{f}(X)$. Intermediate results of a linear algorithm are also computed with a linear algorithm. Hence, it is legitimate to apply the correcting process to such intermediate results.

For example, the previous function g satisfies $g(x,y,z) = f_2(x,y,x) + y^2 + z^2$, with function f_2 defined in Section 3.4. As $\overline{f_2(2^{50}, 2^{25}, 2^{50})} = -2^{50}$,

$$\overline{g_2(x,y,z)} = \overline{f_2(x,y,x)} + y^2 + z^2$$

yields the corrected result

$$\overline{g_2(2^{50}, 2^{25}, 1)} = 1 = g(2^{50}, 2^{25}, 1).$$

In this example, we observe that no other correction than this intermediate correction is necessary to avoid the inaccuracy of the computed result. The final accuracy of this computation relies on the accuracy of this intermediate computation. We call such an intermediate variable a *sensitive intermediate variable*. Of course, to identify sensitive intermediate variables is generally a difficult task.

Correcting sensitive intermediate results extends the range of application and the efficiency of the CENA method. An interesting application of such intermediate correction arises when the stability of the algorithm depends on the accuracy of intermediate results. For example, the computation with Newton's iteration of a multiple root of a polynomial is stabilized when the intermediate evaluations of the polynomial and its derivative are corrected with the CENA method. Such significant examples of intermediate correction are presented in [17].

After this presentation of the principles of the CENA method, we detail the computation of the correcting term $\widehat{\Delta_L}$ in next Sections 4 and 5.

4 Computing the Elementary Rounding Errors

We extend the notation δ_k introduced in relation (3) to define the operation and the operands concerned by the elementary rounding error. For $\circ \in \{+, -, \times, /, \sqrt{}\}$, let $\widehat{x}_i$, $\widehat{x}_j$ and $\widehat{x}_k$ be in $\mathbb{F}$ and such that $\widehat{x}_k = fl(\widehat{x}_i \circ \widehat{x}_j)$. The *absolute elementary rounding error* in the computation of $\widehat{x}_k$ is

$$\delta_k[\circ, x_i, x_j] = fl(\widehat{x}_i \circ \widehat{x}_j) - (\widehat{x}_i \circ \widehat{x}_j) = \delta_k,$$

where $\widehat{x}_j \neq 0$ when $\circ = /$, and $\widehat{x}_i = 0$, $\widehat{x}_j \geq 0$ when $\circ = \sqrt{}$. The elementary rounding error $\delta_k[\circ, x_i, x_j]$ satisfies the two following properties.

Proposition 1 *For $\circ \in \{+, -, \times\}$, the elementary rounding error $\delta_k[\circ, x_i, x_j]$ belongs to $\mathbb{F}$ and is computable using the operations defined on $\mathbb{F}$.*

As the elementary rounding error $\delta_k[\circ, x_i, x_j] \notin \mathbb{F}$ when $\circ \in \{/, \sqrt{}\}$, we introduce the approximate elementary error $\widehat{\delta_k}[\circ, x_i, x_j]$ and an approximation bound β_k, to derive a similar property.

Proposition 2 *For $\circ \in \{/, \sqrt{}\}$, an approximate elementary error $\widehat{\delta_k}[\circ, x_i, x_j] \in \mathbb{F}$ and an approximation bound $\beta_k \in \mathbb{F}$ such that*

$$\left| \widehat{\delta_k}[\circ, x_i, x_j] - \delta_k[\circ, x_i, x_j] \right| \leq \mathbf{u}\, \beta_k,$$

are computable using the operations defined on $\mathbb{F}$.

Table 3: Elementary rounding errors relations, bounds and overheads.

$\circ$	$\delta_k[\circ, y, z]$ for $x_k = y \circ z$	β_k	$t_{\delta_k[\circ]}/t_\circ$
$\pm$	$\mp\,(y - x_k)$ where exponent$(y) \leq$ exponent(z)	0	2
$\times$	$x_k - (y^U \times z^U)\,[(y^U \times z^L) + (y^L \times z^U)] - (y^L \times z^L)$, with $x^U = [x - (x - M)] + (x \times M)$, $x^L = x - x^U$, $(x = y, z)$, and $M = 2^{p_1} + 1$.	0	13
$/$	$\{(x_k \times z) - y - \delta[\times, (y/z), z]\}\,/z$	$\delta_k[/, y, z]$	≈ 3
$\sqrt{}$	$(x_k \times x_k) - y + \delta[\times, x_k, x_k]$	$\frac{5}{2}\,\delta_k[\sqrt{}, y]$	≈ 1.5

The operations $\circ \in \{+, -, \times\}$ satisfy Property 2 with $\beta_k = 0$. So, we can compute the elementary rounding errors δ_k, and the bound β_k for the rounding errors in the computation of δ_k defined by relation (12).

Remark about the proof of Properties 1 and 2: Table 3 summarizes the computing relations for δ_k, the bounds β_k and the overhead introduced by the computation of δ_k assuming an optimal binary arithmetic. No hat notation is used in this table as operands and operations are obviously computed quantities. Elementary rounding errors were largely studied in the 1970s. Except for the square root operator, the relations for computing δ_k are quoted from references [8], [15], [29]. The integer p_1 is such that $p_1 = p/2$ for an even number of digits p of the floating point mantissa, and $p_1 = (p-1)/2$ otherwise.

The numerator of $\delta_k[/, x_i, x_j]$ belongs to $\mathbb{F}$ [30]. Hence, the error bound β_k for this operation only takes into account the rounding error of the last division by x_j in the computation of $\delta_k[/, x_i, x_j]$. For the square root, we neglect the second-order terms in $\mathbf{u}$ and derive the bound β_k for $\delta_k[\sqrt{}, x_i, x_j]$ after some straightforward manipulations (see [18] for details).

In practice, the computation of the elementary errors partly contributes to the time overhead introduced by the CENA method. For each elementary operation $\circ$, let $t_{\delta_k[\circ]}$ be the time to compute the elementary rounding error $\delta_k[\circ]$, and $t_\circ$ be the time to compute the operation $\circ$. As the latter depends on the computer, we assume that the floating point

operations satisfy the following (realistic) relative performances : $t_{\pm} = t_{\times} = t, t_{/} = 10t$ and $t_{\surd} = 20t$ [33]. Hence, the ratios of time $t_{\delta_k[\circ]}/t_{\circ}$ presented in Table 3 measure the time overheads due to the computation of each elementary rounding error.

5 Computing the Partial Derivatives

The partial derivatives $\partial \widehat{f}/\partial \delta_k$ in the correcting factor $\widehat{\Delta_L}$ are computed with automatic differentiation. The automatic differentiation of $\widehat{f}$ is a direct application of the differentiation chain rule

$$\frac{\partial \widehat{f}}{\partial \widehat{x}_k} = \sum_{C \in \Gamma_k} \prod_{\text{arc}(\widehat{x}_i,\, \widehat{x}_j) \in C} \frac{\partial \widehat{x}_j}{\partial \widehat{x}_i} \quad (1 \leq k \leq N).$$

This computation relies on a graph representation of algorithm $\widehat{f}$. The graph vertices are the initial and intermediate variables $\widehat{x}_k$ of $\widehat{f}$ $(1 \leq k \leq N)$. The graph arc between $\widehat{x}_i$ and $\widehat{x}_j$, $\text{arc}(\widehat{x}_i, \widehat{x}_j)$, is weighted by $\partial \widehat{x}_j/\partial \widehat{x}_i$ for $i < j$. This elementary partial derivative $\partial \widehat{x}_j/\partial \widehat{x}_i$ is computed given that $\widehat{x}_j = fl(\widehat{x}_i \circ \widehat{x}_l)$ $(i \leq l < j)$. For a given intermediate variable $\widehat{x}_k$, Γ_k is the set of paths C from $\widehat{x}_k$ to the result $\widehat{x}_N = \widehat{f}(X)$ $(n+1 \leq k \leq N)$. Let us introduce a Vertices_From function that returns the (at most two) vertices connected by an arc to a given vertices. Vertices_From$(\widehat{x}_k)$ returns $\widehat{x}_i$ and $\widehat{x}_j$ when $\widehat{x}_k = fl(\widehat{x}_i \circ \widehat{x}_j)$.

As $\widehat{x}_k = x_k + \delta_k$,

$$\frac{\partial \widehat{x}_k}{\partial \delta_k} = 1 \quad (n+1 \leq k \leq N),$$

and as $x_k \in \mathbb{F}$ for $1 \leq k \leq n$,

$$\frac{\partial \widehat{x}_k}{\partial \delta_k} = 0 \quad (1 \leq k \leq n).$$

So the expected partial derivatives with respect to the elementary rounding errors are computable using

$$\frac{\partial \widehat{f}}{\partial \delta_k} = \sum_{C \in \Gamma_k} \prod_{\text{arc}(\widehat{x}_i,\, \widehat{x}_j) \in C} \frac{\partial \widehat{x}_j}{\partial \widehat{x}_i} \quad (n+1 \leq k \leq N). \tag{15}$$

The *reverse mode* of automatic differentiation allows us to compute relation (15). The reverse mode includes two stages [14]. The graph construction is first performed, for example as algorithm $\widehat{f}$ is processed, from the bottom to the top, that is, from $\widehat{x}_1, \ldots, \widehat{x}_n$ to $\widehat{x}_N$. Then, the summation in relation (15) is taken from the top to the bottom of the graph, that is, from $\widehat{x}_N$ to $\widehat{x}_k$ with $n+1 \leq k < N$.

Algorithm 5 computes $\partial \widehat{f}/\partial \delta_k$ by automatic differe ntiation in reverse mode.

Require: $\partial x_k/\partial x_i\,,\ k = n+1, N$
  for $k = 1$ **to** $N-1$ **do**
    $Dx_k = 0$
  end for
  $Dx_N = 1$
  for $k = N$ **to** $n+1$ **do**
    for $i \in$ Vertices_From(x_k) **do**
      $Dx_i = Dx_i + Dx_k \times (\partial x_k/\partial x_i)$
    end for
  end for
  $\partial \widehat{f}/\partial \delta_k = Dx_k$

Assuming the graph has been constructed, the computation of the derivatives using automatic differentiation in reverse mode is described with Algorithm 5.

Using running error analysis, we bound the rounding errors introduced by the computation of the derivatives $\partial \widehat{f}/\partial \delta_k$, and hence derive the bound α_k defined by relation (11) for $n+1 \leq k \leq N$.

The bound analysis is similar to the analysis conducted for E_C in Section 3.4. The rounding errors in the summation and the product of Algorithm 5 are taken into account using the relations (1) and (2) of the standard model of floating point arithmetic. The computation of $\partial \widehat{x}_k/\partial \widehat{x}_i$ may suffer from rounding error when $\widehat{x}_k = \widehat{x}_j/\widehat{x}_i$ or $\widehat{x}_k = \sqrt{\widehat{x}_i}$, as $(-1/\widehat{x}_i^2)$ and $1/(2\,\widehat{x}_k)$ are computed. These rounding errors are taken into account within the global error of the computation of (15), introducing a scalar C_{ki} and a bound γ_{ki} such that

$$C_{ki} = \frac{\partial \widehat{x}_k}{\partial \widehat{x}_i},$$

and

$$\left| \widehat{C}_{ki} - \frac{\partial \widehat{x}_k}{\partial \widehat{x}_i} \right| \leq \mathbf{u}\,\gamma_{ki}.$$

In optimal binary arithmetic, we have

$$\gamma_{ki} = \begin{cases} 2.02\,|\,\widehat{C}_{ki}| & \text{when } \widehat{x}_k = \widehat{x}_j/\widehat{x}_i, \\ |\,\widehat{C}_{ki}| & \text{when } \widehat{x}_k = \sqrt{\widehat{x}_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The first result derives from a classic *a priori* rounding error analysis of the computation of $(-1/\widehat{x}_i^2)$ (for example, applying Lemma 3.4 of [12] as $2\mathbf{u} < 1$). The second result is similar for $1/(2\,\widehat{x}_k)$ and uses the fact that the base $\mathsf{b} = 2$. The same kind of bounds could be derived according to other specific floating point arithmetic properties. Running error analysis yields here equivalent bound with more operations to compute.

To simplify the algorithm of running error bound associated with the computation of the derivatives, the second-order terms in $\mathbf{u}$ are hereafter neglected. Taking them into account leads to an algorithm similar to Algorithm 4. Thus, Algorithm 6 computes both the derivatives $\partial\widehat{f}/\partial\delta_k$ and the running error bound α_k associated to the rounding errors introduced by this computation.

Algorithm 6 computes $\partial\widehat{f}/\partial\delta_k$ and bound α_k.

Require: $\partial x_k/\partial x_i\ ,\ k = n+1, N$
  for $k = 1$ **to** $N-1$ **do**
    $Dx_k = 0$
    $e_k = 0$
  end for
  $Dx_N = 1$
  for $k = 1$ **to** N **do**
    $\alpha_k = 0$
  end for
  for $k = N$ **to** $n+1$ **do**
    for $i \in$ Vertices_From(x_k) **do**
      $C_{ki} = \partial x_k/\partial x_i$
      $Dx_i = Dx_i + Dx_k \times C_{ki}$
      $e_i = e_i + |Dx_k \times C_{ki}| + |Dx_i| + e_k \times |C_{ki}| + \gamma_{ik} \times |Dx_k|$
    end for
  end for
  $\partial\,\widehat{f}/\partial\delta_k = Dx_k$
  $\alpha_k = \mathbf{u} \times e_k$

The computation of the derivatives is the main contribution to the time overhead introduced by the CENA method we began to discuss at the end of Section 4.

The main theoretical results about time and space complexity of automatic differentiation are from [4]. For the reverse mode, the relative time overhead is bounded by m times a constant when $\widehat{f}(X) \in \mathbb{R}^m$. For straightforward implementation, Iri derives the constant 4 in [14]. The relative space overhead in reverse mode is bounded by a constant times the sum of $\widehat{f}$ computation space and time units.

The practical overheads of existing automatic differentiation tools are not necessarily of the same order. The graph construction mainly consists of memory management that is very sensitive to the size of $\widehat{f}$, implementation choices and machine memory characteristics. In our experience, a constant of an order of 20 is a more reasonable practical bound for the relative time overhead. It is interesting to note than Linnainmaa's pioneering work reports that the linearization error increased the execution time by a factor of a few tens [23].

6 The CENA Method

We summarize the previous results and give a description of the three main stages of the CENA method. Then, we emphasize the difference between the CENA method and increasing the precision.

6.1 The Steps of the CENA Method

1. Before computing $\widehat{f}$: Choose the data $X = (x_1, \ldots, x_n) \in \mathbb{F}^n$ and the result variable x_N (in the scalar case) that defines $\widehat{x}_N = \widehat{f}(X)$ for $\widehat{f}$ satisfying Definition 1.

2. During the computation of $\widehat{f}$: For each elementary computation $\widehat{x}_k = fl(\widehat{x}_i \circ \widehat{x}_j)$,

   - Compute $\widehat{x}_k$;
   - Compute the elementary error $\delta_k[\circ, x_i, x_j]$ according to the relations shown in Table 3;
   - Compute the associated rounding error bound β_k according to the relations shown in Table 3;
   - Compute the elementary partial derivatives $\partial\widehat{x}_k/\partial\widehat{x}_i$ and $\partial\widehat{x}_k/\partial\widehat{x}_j$, given that $\widehat{x}_k = fl(\widehat{x}_i \circ \widehat{x}_j)$;
   - Compute the associated rounding error bound γ_{ki} and γ_{kj} according to relation (16);
   - Update $\widehat{f}$ computational graph such that function Vertices_From($\widehat{x}_k$) returns $\widehat{x}_i$ and $\widehat{x}_j$;
   - Update $\widehat{f}$ computational graph storing the values of $\delta_k[\circ, x_i, x_j]$, β_k, $\partial\widehat{x}_k/\partial\widehat{x}_i$, $\partial\widehat{x}_k/\partial\widehat{x}_j$, γ_{ki} and γ_{kj}.

3. After $\widehat{x}_N = \widehat{f}(X)$ is computed :

   - Compute the correcting factor $\widehat{\Delta_L}$ according to relation (5);
   - Compute the corrected result $\overline{f(X)}$ according to relation (7);
   - Compute the error bound α_k for the computation of the derivatives according to Algorithm 6;
   - Compute the confidence threshold B_{E_C} according to relation (13) and Algorithm 4;
   - (optional) Assess the accuracy of the original computed result $\widehat{x}_N$ according to Corollary 2.

6.2 The CENA Method *versus* Increasing the Precision

The computation of the elementary rounding errors (Table 3) might suggest that the CENA method is tantamount to double precision computation. This is not the case. The computation of $\widehat{x}_k$ generates an absolute rounding error that is small with respect to $\widehat{x}_k$ as $|\delta_k| \leq \mathbf{u}\,|\widehat{x}_k|$. When the first-order term in δ_k of the global forward error is not small with respect to $\widehat{x}_N$, δ_k may significantly contribute to the inaccuracy of the computed result $\widehat{x}_N$. Computing x_N in precision $\mathbf{u}$ introduces terms in δ_k in the global forward error of $\widehat{x}_N$.

Increasing the precision $\mathbf{u}$ provides a smaller δ_k and may yield $\delta_k = 0$. In this case, the double precision result may be more accurate that the original one since no term in δ_k is introduced. But if $\delta_k \neq 0$, the previously described first-order effect of δ_k on the accuracy of $\widehat{x}_N$ is still valid. Increasing the precision only delays the influence of δ_k.

With the CENA method, we compute this elementary error δ_k and subtract its first-order effect in the final result $\widehat{x}_N$ at the end of the computation. If the global forward error for $\widehat{x}_N$ mainly depends on the first-order term in δ_k, as for example for linear algorithms, then the corrected result provides a more accurate result than the original $\widehat{x}_N$ as it is free of the first-order effect of the δ_k. Moreover, the corrected result with the CENA method is more accurate than the result of any higher precision computation such that the elementary errors δ_k do not vanished or compensated each other.

For the previous function $f(x, y, z) = x^2 - y^2 - z^2$, let $\widehat{f}_3$ be the (linear) Algorithm 2 evaluated in IEEE-754 double precision ($\mathbf{u} = 2^{-53}$) [13]. We have

$$\widehat{f}_3(2^{25}, 1, 2^{25}) = -1 = f(2^{25}, 1, 2^{25}),$$

which is now exact but

$$\widehat{f}_3(2^{27}, 1, 2^{27}) = 0,$$

that illustrates the delay of the influence of the δ_k. Hence, the global forward errors for the computations in single and double precision of algorithm $\widehat{f}_2$ are equal. Correcting the linear algorithm $\widehat{f}_2$ still provides using single precision

$$\overline{f_2(2^{27}, 1, 2^{27})} = -1 = f(2^{27}, 1, 2^{27}).$$

Thus, the corrected result is the exact result and this will be true for any admissible $(x, y, z) \in \mathbb{F}$.

7 Conclusion

Finite precision computation provides results that suffer from every elementary rounding errors coming from floating point arithmetic operations. For linear algorithms, this global forward error is a linear function of the elementary rounding errors. The CENA method computes this global forward error and provides an accurate corrected result.

The correction is computed using automatic differentiation in reverse mode and elementary errors computation. Such computations also contain rounding errors whose effect on the correcting term is taken into account with the residual error bound B_{E_C}. The computation of this bound uses mainly running error analysis and *a priori* approximations.

We apply the CENA method to correct final results or sensitive intermediate variables computed with linear algorithms. The corrected result improves the accuracy of the computed result, and sometimes allows us to prove that this computed result is inaccurate to all figures. We have proven that the CENA method is not equivalent to increase the precision of the computation.

The efficiency and the reliability of the CENA method are illustrated in [17] with the accuracy improvement of more realistic numerical algorithms as inner product computation and triangular linear system solving. When rounding errors introduce algorithm instability, intermediate correction may produce stable and reliable computations. For example, the CENA method stabilizes Newton's iteration applied to compute the multiple root of a polynomial [17].

The CENA method is a forward deterministic method, useful to highlight and correct when it is possible, the unavoidable effect of rounding errors in scientific computation algorithms.

The reliability of the CENA method depends on the linearity property of the analyzed algorithm. Thus, its application is not universally efficient. Numerical applications frequently use elementary functions such as trigonometric functions, exponential functions, ... Further development of the CENA method should efficiently manage the computation of accurate approximations of the rounding error of elementary functions. In practice, the performance of the implementation should also take into account the newest improvements in automatic differentiation software.

Acknowledgments: The author is grateful to Thierry Braconnier and Fabrice Nativel (Université de La Réunion) for their collaboration to parts of this work, and thanks Nicholas J. Higham (University of Manchester) for his several useful suggestions, Sheung H. Cheng and Philip I. Davies (University of Manchester) for their numerous language corrections.

References

[1] WWW resources about interval arithmetic. URL = `http://interval.usl.edu/`.

[2] Göltz Alefeld and Jürgen Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[3] Jean-Claude Bajard, Olivier Beaumont, Jean-Marie Chesneaux, Marc Daumas, Jocelyne Erhel, Dominique Michelucci, Jean-Michel Muller, Bernard Philippe, Nathalie

Revol, Jean-Louis Roch, and Jean Vignes. *Quality of Computers Computations : Toward More Reliable Arithmetics ?* Masson, Paris, 1997. (In French).

[4] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[5] Françoise Chaitin-Chatelin and Valérie Frayssé. *Lectures on Finite Precision Computations.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[6] Jean-Marie Chesneaux. *Stochastic Arithmetic and CADNA software.* Habilitation à Diriger des Recherches Thesis, Université Pierre et Marie Curie, Paris, France, November 1995. (In French).

[7] Jean-Marie Chesneaux and Jean Vignes. Les fondements de l'arithmétique stochastique. *C. R. Acad. Sci. Paris Sér. I Math.*, 315(13):1435–1440, 1992.

[8] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18:224–242, 1971.

[9] Jocelyne Erhel. Experiments with data pertubations to study condition numbers and numerical stability. *Computing*, 51(1):29–44, 1993.

[10] Philippe Francois. *Contribution à l'étude de méthodes de contrôle automatique de l'erreur d'arrondi : la méthodologie SCALP.* PhD thesis, Institut National Polytechnique, Grenoble, France, December 1989. (In French).

[11] Peter Henrici. *Elements of Numerical Analysis.* Wiley, New York, 1964.

[12] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[13] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.* Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

[14] Masao Iri. History of automatic differentiation and rounding error estimation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 3–16. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991.

[15] Donald E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms.* Addison-Wesley, Reading, MA, USA, third edition, 1998.

[16] Ulrich W. Kulisch and Willard L. Miranker. *Computer Arithmetic in Theory and in Practice.* Academic Press, New York, 1981.

[17] Philippe Langlois. Final and intermediate correction with the CENA method. Research Report, Institut National de Recherche en Informatique et Automatique, Rocquencourt, France, November 1999. (in progress).

[18] Philippe Langlois and Fabrice Nativel. Reduction and bounding of the rounding error in floating point arithmetic. *C.R. Acad. Sci. Paris, Série 1*, 327:781–786, 1998.

[19] Philippe Langlois and Fabrice Nativel. When automatic linear correction of rounding errors is exact. *C.R. Acad. Sci. Paris, Série 1*, 328:543–548, 1999. Erratum in 328:829, 1999.

[20] John L. Larson and Ahmed H. Sameh. Efficient calculation of the effects of roundoff errors. *ACM Trans. Math. Software*, 4(3):228–236, 1978. Errata 5(3):372, 1979.

[21] Seppo Linnainmaa. *The Representation of Cumulative Rounding Error of an Algorithm as a Taylor Expansion of Local Rounding Errors.* PhD thesis, University of Helsinki, Helsinki, Finland, 1972. (In Finnish, cited in [22]).

[22] Seppo Linnainmaa. Towards accurate statistical estimation of rounding errors in floating-point computations. *BIT*, 15:165–173, 1975.

[23] Seppo Linnainmaa. Error linearization as an effective tool for experimental analysis of the numerical stability of algorithms. *BIT*, 23:346–359, 1983.

[24] Webb Miller. Toward mechanical verification of properties of roundoff error propagation. In *Proceedings of Fifth Annual ACM Symposium on Theory on Computing*, pages 50–58. ACM, New-York, USA, 1973.

[25] Webb Miller and Celia Wrathall. *Software for Roundoff Analysis of Matrix Algorithms.* Academic Press, New York, 1980.

[26] Ramon E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1966.

[27] Fabrice Nativel. *Fiabilité numérique et précision finie : une méthode automatique de correction linéaire de l'erreur d'arrondi.* PhD thesis, Université de La Réunion, Saint-Denis de La Réunion, France, December 1998. (In French).

[28] G. Peters and J. H. Wilkinson. Practical problems arising in the solution of polynomial equations. *J. Inst. Maths Applics*, 8:16–35, 1971.

[29] Michèle Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numer. Math.*, 19:400–406, 1972.

[30] Michèle Pichat. *Contribution à l'étude des erreurs d'arrondi en arithmétique à virgule flottante.* Doctorat d'Etat Thesis, Université de Grenoble 1, Grenoble, France, 1976. (In French).

[31] Michèle Pichat and Jean Vignes. *Ingénierie du contrôle de la précision des calculs sur ordinateur*. Technip, Paris, 1993. (In French).

[32] Michel La Porte and Jean Vignes. Etude statistique des erreurs dans l'arithmétique des ordinateurs ; application au contrôle des résultats d'algorithmes numériques. *Numer. Math.*, 23:63–72, 1974. (In French).

[33] Peter Soderquist and Miriam Leeser. Division and square root: Choosing the right implementation. *IEEE Micro*, 17(4):56–66, 1997.

[34] David R. Stoutemyer. Automatic error analysis using computer algebraic manipulation. *ACM Trans. Math. Software*, 3(1):26–43, 1977.

[35] Martti Tienari. A statistical model of roundoff error for varying length floating-point arithmetic. *BIT*, 10:355–365, 1970.

[36] Jean Vignes. A stochastic approach to the analysis of round-off error propagation – a survey of the Cestac method. In *Proceedings of Second Real Numbers and Computer Conference*, pages 233–251. Marseille, 1996.

[37] James H. Wilkinson. Error analysis revisited. *IMA Bulletin*, 22(11/12):192–200, 1986.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399